

# Low-Density MDS Codes and Factors of Complete Graphs

Lihao Xu, Vasken Bohossian, Jehoshua Bruck, *Senior Member, IEEE*, and David G. Wagner

**Abstract**—We present a class of array code of size  $n \times l$ , where  $l = 2n$  or  $2n + 1$ , called **B-Code**. The distances of the B-Code and its dual are 3 and  $l - 1$ , respectively. The B-Code and its dual are optimal in the sense that i) they are maximum-distance separable (MDS), ii) they have an optimal encoding property, i.e., the number of the parity bits that are affected by change of a single information bit is minimal, and iii) they have optimal length. Using a new graph description of the codes, we prove an equivalence relation between the construction of the B-Code (or its dual) and a combinatorial problem known as *perfect one-factorization* of complete graphs, thus obtaining constructions of two families of the B-Code and its dual, one of which is new. Efficient decoding algorithms are also given, both for erasure correcting and for error correcting. The existence of perfect one-factorizations for every complete graph with an even number of nodes is a 35 years long conjecture in graph theory. The construction of B-Codes of arbitrary odd length will provide an affirmative answer to the conjecture.

**Index Terms**—Array codes, low density, MDS codes, perfect one-factorization, update complexity.

## I. INTRODUCTION

ARRAY codes have important applications in communication and storage systems [6], [7], and have been studied extensively [2]–[4], [9]. Array codes are a class of linear codes, where information and parity bits are placed in a two-dimensional (or multidimensional) array rather than a one-dimensional vector. The information and parity bits are defined over an Abelian group  $G(q)$  with an addition operation  $+$ . Usually  $q = 2$ , i.e., the bits are just binary bits and  $+$  is just the simple *XOR* (*exclusive-OR*). (For simplicity, we will assume  $q = 2$  in this paper. However, most results in this paper still hold for arbitrary  $q$ .) On the other hand, an array code of size  $n \times l$  (the array has  $l$  columns, each of which has  $n$  bits) can also be regarded as a one-dimensional code defined over the Abelian group  $G(q^n)$ , and the distance  $d$  of the code can also be defined over  $G(q^n)$ . Let  $N$  be the number

of its codewords, then the dimension  $k$  of the array code can be defined to be  $k = \log_{q^n} N$ , as for usual one-dimensional codes. So the above array code can also be viewed as an  $(l, k, d)$  code over  $G(q^n)$ , which corresponds to the error model that will be used in this paper: if one bit in a column is an error or an erasure, then this column is considered to be an error or an erasure. If the equality of the Singleton bound [12] holds for the above  $(l, k, d)$ :  $d = l - k + 1$ , then the array code is called an MDS (*Maximum-Distance Separable*) array code.

A common property of array codes is that the encoding and decoding procedures use only simple XOR's, thus MDS array codes are more efficient in terms of computation complexity than Reed–Solomon codes, since the encoding and decoding of Reed–Solomon codes need complex finite-field operations [6].

One important parameter of array codes is the average number of parity bits affected by a change of a single information bit in the codes, called the *update complexity* in this paper. The update complexity is particularly crucial when the codes are used in storage applications that update information frequently. It also measures the encoding complexity of the code. The lower this parameter is, the simpler the encoding operations are. If a code is described by a *parity-check matrix*, then this parameter is the average *row density*—the number of nonzero entries in a row—of the parity-check matrix. Research has been done to reduce this parameter or to make the density of parity-check matrix of codes as low as possible [8], [13]. The update complexity of *EVENODD* codes approaches 2 as the length (number of the columns) of the codes increases. But it was proven in [3] that for any linear array code of distance 3 with separate information and parity columns, the update complexity is always *strictly* larger than 2 (the obvious lower bound). Then a natural question is whether the update complexity of 2 is achievable for general array codes of distance 3. A positive answer to the foregoing question was given more than a decade ago [18], and the code in [18] was described by its *parity-check matrix* with lower density and represented recently in a general form, also by parity-check matrix, in [5]. Recently, another class of MDS array codes with this property was also found using geometrical construction [17].

In this paper, we describe a class of MDS array codes of distance 3, whose update complexity is *optimal*, i.e., 2. The code is called the B-Code in this paper. The B-Code is of size  $n \times l$ , where  $l = 2n$  or  $2n + 1$ . The B-Code not only includes the family of array codes in [18], but also gives a new family of MDS array codes that was not known before. Its dual is also an MDS array code of distance  $l - 1$ , with optimal update

Manuscript received August 25, 1998. This work was supported in part by the NSF Young Investigator Award CCR-9457811, by an IBM Partnership Award, by the Sloan Research Fellowship, and by DARPA through an agreement with NASA/OSAT.

L. Xu is with the Department of Computer Science, Washington University, St. Louis, MO 63130 USA (e-mail: lihao@cs.wustl.edu).

V. Bohossian and J. Bruck are with the California Institute of Technology, Mail Code 136-93, Pasadena, CA 91125 USA (e-mail: {vincent,bruck}@paradise.caltech.edu).

D. G. Wagner is with the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1 (e-mail: dgwagner@math.uwaterloo.ca).

Communicated by A. M. Barg, Associate Editor for Coding Theory.

Publisher Item Identifier S 0018-9448(99)06065-4.

TABLE I  
B-CODE VERSUS REED-SOLOMON AND EVENODD

Codes \ Properties	MDS	XOR	Optimal Update	Optimal Length	Balanced Computation
Reed Solomon	Yes	No	No	Yes	No
EVENODD	Yes	Yes	No	No	No
B	Yes	Yes	Yes	Yes	Yes

$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$
$a_2 + a_3$	$a_3 + a_4$	$a_4 + a_5$	$a_5 + a_6$	$a_6 + a_1$	$a_1 + a_2$
$a_4 + a_6$	$a_5 + a_1$	$a_6 + a_2$	$a_1 + a_3$	$a_2 + a_4$	$a_3 + a_5$

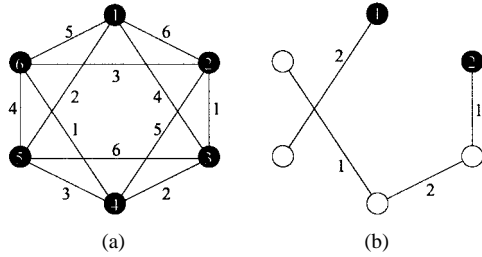


Fig. 1.  $\hat{B}_6$ , the dual B-Code of length 6, is a  $3 \times 6$  MDS array code or a  $(6, 2, 5)$  MDS code over  $G(2^3)$ . The  $a_i$ 's are the information bits. (a) The graph representation of  $\hat{B}_6$ . (b) A decoding path for the erasure of columns 3–6 (i.e., only columns 1 and 2 are available).

complexity as well. An algebraic definition of the B-Code and its dual will be given later in the next section.

The novelty of this paper is to use a graph approach to describe the B-Code and its dual, making the design of the code easier and more direct. Fig. 1(a) shows  $\hat{B}_6$ , the dual B-Code of length 6. In addition to the usual representation of a code as an array of information and parity bits, the B-Code can be represented by a labeled graph in which every vertex corresponds to an information bit and each edge represents a parity bit: each parity bit is simply the sum of the two information bits that constitute the edge. The edges and vertices of the graph are labeled with a column index: the  $i$ th column of the code consists of the information bit  $a_i$  and the parity bits with the column index  $i$ . The same notation will be used hereafter in this paper.  $\hat{B}_6$  has distance 5 and can, therefore, tolerate any erasure of four columns. Fig. 1(b) shows a decoding path for the erasure of columns 3–6. Use  $a_2$  (from column 2) together with parity  $a_2 + a_3$  (from column 1) to recover  $a_3$ . Use the latter along with parity  $a_3 + a_4$  (from column 2) to recover  $a_4$ , etc. For any four-columns erasure, such a decoding path exists.

By using this new graph description, it will be proven in this paper that constructing a B-Code (and its dual code) is equivalent to a three-decade old graph theory problem, the *perfect one-factorizations* of complete graphs [15], denoted *P1F*. Using results on *P1F*, we can construct two infinite families of B-Codes, one of which can be shown to be the construction of [18]. In addition, there are a number of values for which *P1F*'s exist that are not in the two infinite families; these result in constructions of the B-Codes of all lengths up to 49. The existence of perfect one-factorizations for every complete graph with an even number of nodes is a 35-year-old conjecture in graph theory. An affirmative answer to this

conjecture will provide the B-Code constructions of arbitrary length. Alternately, the construction of the B-Codes of arbitrary odd length will provide an affirmative answer to the conjecture.

As already proven in [18], the B-Code achieves the *maximum* length that MDS codes with optimal update property can have, thus the B-Code has *optimal length*, twice of that of the code in [17] with a same column size. In addition, since the parity bits are evenly distributed over all columns, and each parity bit requires the same amount of XOR operations, the computation complexity for computing parity bits is balanced, i.e., the B-Code features *balanced computation* as well. The properties of the B-Code are summarized in Table I, together with a comparison with Reed–Solomon and EVENODD codes.

The main contributions of this paper are as follows:

- 1) proving the *equivalence* of the perfect one-factorization of complete graphs and the MDS code constructions;
- 2) providing *constructions* for a new family of low-density MDS array codes;
- 3) proving that, in general, the dual of an MDS array code is still MDS.

The paper is organized as follows. In Section II, we describe the B-Code and its dual using a new graph representation. In Section III, we reveal the relation between the B-Code and the *P1F* problem. We also give efficient *erasure* and *error* decoding algorithms for the B-Code. In Section IV, we further discuss the equivalence between the B-Code and *P1F*. In Section V, we conclude the paper and present some future research directions.

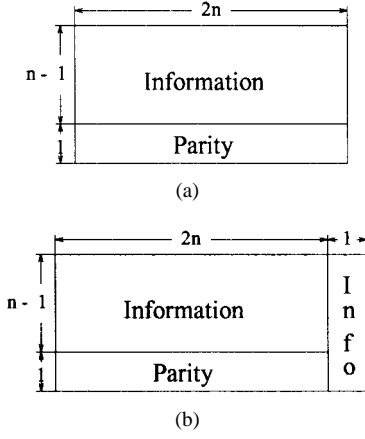
## II. B-CODE AND ITS DUAL

As already described, a B-Code is an MDS code of size  $n \times l$ , with distance 3. The MDS property of the B-Code implies that out of  $nl$  bits, exactly  $2n$  bits should be parity bits. In this section, in addition to an algebraic definition of the B-Code and its dual, we describe the codes using graphs. We also prove that, in general, the dual of an MDS array code is also MDS.

### A. Definition and Structure of the B-Code

We first give an algebraic definition of the B-Code and its dual. The details of representing general array codes using *parity-check* matrix or *generator* matrix will be discussed in the following subsection.

**Definition 1 (B-Code and Dual B-Code):** Let  $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_l\}$  be a binary matrix, where  $l = 2n$  or  $l = 2n + 1$ , and  $\mathbf{h} = \{h_{i,j}\}$  is a binary submatrix of size  $2n \times n$ , for  $1 \leq i \leq 2n$ ,  $1 \leq j \leq n$ , and  $1 \leq k \leq l$ . If  $\mathbf{H}$  meets following conditions:

Fig. 2. Structures of (a)  $B_{2n}$  and (b)  $B_{2n+1}$ .

- 1)  $h_{i,ni} = 1$  ( $1 \leq i \leq 2n$ );
- 2) the weight (number of 1's) of each row of  $\mathbf{H}$  is  $l-1$ ;
- 3) for any  $m$  and  $k$ , where  $1 \leq m, k \leq l$  and  $m \neq k$ , the square matrix  $\{h_m, h_k\}$  is nonsingular;

then the code is called the B-Code if  $\mathbf{H}$  is its parity-check matrix, denoted  $B_l$ ; and the code is called the dual B-Code if  $\mathbf{H}$  is its generator matrix, denoted  $\hat{B}_l$ .

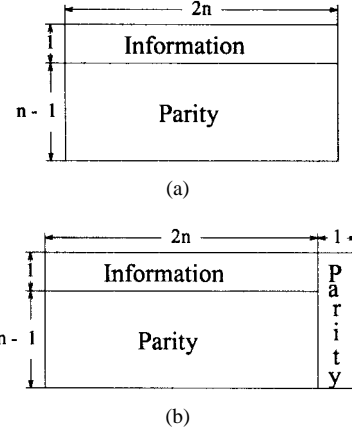
Such a matrix and the corresponding codes can be found in the following subsection. The codes according to the above definition are difficult to construct. We will soon give a new description of the codes using graphs, which makes the construction much easier.

In structure, the B-Code is an array code of size  $n \times l$ , i.e., with  $n$  rows and  $l$  columns. For  $B_{2n}$ , the first  $n-1$  rows are information rows, and the last row is a parity row, i.e., all the bits in the first  $n-1$  rows are information bits, while the  $2n$  bits in the last row are parity bits. The structure of  $B_{2n+1}$  can be derived from that of  $B_{2n}$  simply by adding one more information column as the last column. Their structures are shown in Fig. 2.

Intuitively, if the roles of the information and parity bits of the B-Code are exchanged, i.e., the parity bits are placed in the entries which originally were for the information bits and vice versa, then we get the dual code of the B-Code for length  $l$ . We will soon give a more rigorous definition of the dual code for general array codes, and prove that the dual of a general MDS array code is also MDS. In particular, the dual B-Code is also an MDS array code; it has distance  $l-1$ , i.e., the dual B-Code can be recovered from any two of its columns. Fig. 3 shows the structures of  $\hat{B}_{2n}$  and  $\hat{B}_{2n+1}$ .

### B. Dual Array Codes

Array codes are linear codes which can be described by parity-check or generator matrices. Consider an array code of size  $n \times l$  over  $G(q)$ . A codeword of this code can be represented by a vector of length  $nl$  over  $G(q)$ : it consists of  $l$  blocks, each of which includes  $n$  components. The correspondence between the vector description and the array description is obvious: the  $i$ th block of the vector corresponds to the  $i$ th column of the array, and the  $n$  components within a block are just the  $n$  symbols within the corresponding column.

Fig. 3. Structures of (a)  $\hat{B}_{2n}$  and (b)  $\hat{B}_{2n+1}$ .

$a_0$	$a_1$	$a_2$	$a_3$
$p_0$	$p_1$	$p_2$	$p_3$

(a)

$$c = (a_0 \ p_0 \mid a_1 \ p_1 \mid a_2 \ p_2 \mid a_3 \ p_3)$$

(b)

Fig. 4. A codeword of  $2 \times 4$  code in (a) array form and (b) vector form.

A codeword  $c$  of a  $2 \times 4$  array code is shown in both array form and vector form in Fig. 4.

Using this vector form, an array code of size  $n \times l$  with  $nr$  parity bits can be described by its parity-check matrix  $\mathbf{H}$ , of size  $nr \times nl$ , or its generator matrix  $\mathbf{G}$ , of size  $n(l-r) \times nl$ ; here  $r$  is the number of parity (redundant) columns as if some columns consisted of only parity bits. Like for other one-dimensional linear block codes, it is easy to observe that for a codeword  $c$  of the array code and an information vector  $m$  of length  $n(l-r)$ , the identities that  $c = m\mathbf{G}$  and  $c\mathbf{H}^T = \mathbf{0}$  still hold, or equivalently,  $\mathbf{G}\mathbf{H}^T = \mathbf{0}$ . In Fig. 4, let the  $a_i$ 's be information bits and  $p_i$ 's be parity bits. Specifically, when  $p_i = a_{(i+1) \bmod 4} + a_{(i+2) \bmod 4}$ , for  $i = 0, 1, 2, 3$ , we get a B-Code of length 4, i.e.,  $B_4$ , with  $n = 2$ ,  $l = 4$ , and  $r = 2$ . Its parity-check matrix can be described as follows:

$$\mathbf{H} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Accordingly, its generator matrix is as follows:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Using the vector form of array codes, we can define dual of array codes as for a conventional one-dimensional linear block code, i.e.,

**Definition 2 (Dual Array Code):** Let  $C$  be a linear array code of size  $n \times l$  over  $G(q)$ , then its dual code  $C^\perp$  is defined as

$$C^\perp = \{\mathbf{u} \in G(q)^{nl} : \mathbf{u} \cdot \mathbf{v} = 0 \text{ for all } \mathbf{v} \in C\}$$

where  $\cdot$  is the conventional dot product of vectors.

It follows that, as with one-dimensional linear block codes, the parity-check matrix of an array code is the generator matrix of its dual code. One would expect that other properties of dual codes that hold for one-dimensional linear block codes also hold for array codes. In particular, the dual of MDS array code is also MDS. (Reference [5] gives a proof for the above statement, but it implicitly assumes that information bits and parity bits are not mixed in a same column.) However, for general array codes, since information and parity bits can be mixed in the same column, it is not as obvious that this property holds as it seems to be. Fortunately, this property can be generalized to general linear array codes, and we will prove it here.

**Theorem 1:** The dual code of an MDS array code is also MDS.

*Proof:* Consider an MDS array code  $C$  of size  $n \times l$ . Suppose its distance is  $r + 1$  with respect to columns. The parity-check matrix of  $C$  can then be written as  $\mathbf{H} = (h_1 \ h_2 \ \cdots \ h_l)$ , where  $h_i$  is a submatrix of size  $nr \times n$  that corresponds to the  $i$ th block in the vector form of a codeword or to the  $i$ th column in the array ( $1 \leq i \leq l$ ). Since  $C$  is MDS, any combination of  $r$  submatrices ( $h_i$ 's) is linearly independent, in terms of their columns.

Since  $\mathbf{H}$  is the generator matrix of the dual code  $C^\perp$ , let a nonzero codeword  $c \in C^\perp$  have  $s$  nonzero columns, where  $s \leq l - r$ , thus  $c$  has zero-columns in some set of  $r$  blocks  $h_i$ . Without loss of generality, let these blocks be  $(h_1 \ h_2 \ \cdots \ h_r)$ . Since  $c$  is by definition a linear combination of the  $r$  rows of  $\mathbf{H}$  (this still holds for any linear array code), the  $nr \times nr$  square submatrix formed by  $(h_1 \ h_2 \ \cdots \ h_r)$  must be singular, which contradicts the fact that any combination of  $r$  blocks ( $h_i$ 's) are linearly independent. Thus the minimum column weight of any codeword of  $C^\perp$  must be greater than  $l - r$ , i.e., the minimum distance of  $C^\perp$  is greater than  $l - r$ . By the Singleton bound [16], this shows the dual code  $C^\perp$  is also MDS.  $\square$

Since one-dimensional linear block codes are just a special case of array codes, the above theorem certainly holds and the proof above reduces to one of many proofs for one-dimensional block codes [16].

### C. A New Graph Description of the B-Code

Typically, an array code is described by its geometrical construction lines [2]–[4], [9], or by its parity-check (or generator) matrix [5], [18], as was in Section II-A. However, constructions of array codes are usually difficult to get using these descriptions. Here, we describe the B-Code and its dual using a new graph approach. By relating the graph conditions for constructing the B-Code to a classical graph problem, perfect one-factorization of complete graphs, we obtain new constructions.

For any array code, each parity bit is the sum of some information bits; for binary codes, the addition is just the simple XOR (binary *exclusive-OR*) operation. If a parity bit  $P$  is the sum of an information bit  $I$  and other information bits, then we say that the information bit  $I$  *appears* in the parity bit  $P$ . Now consider the dual B-Code  $\hat{B}_l$ . Because of its MDS and optimal encoding properties, each information bit must appear *exactly*  $l - 2$  times in the parity bits. Since the numbers of the total information and parity bits are  $2n$  and  $nl - 2n$ , respectively, each parity bit must be the sum of  $2n(l - 2)/(nl - 2n)$  or *exactly* two information bits. (This is reflected in the parity-check matrix by the fact that the weight of each row is exactly 3.) So if we represent an information bit as a vertex, then a parity bit can be represented by an edge, where the parity bit is the sum of the two information bits whose vertices form the edge. This is the key idea of describing the B-Code and its dual with graphs.

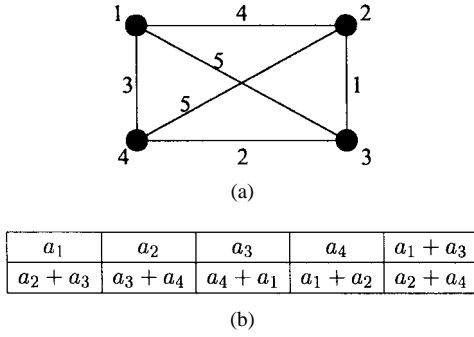
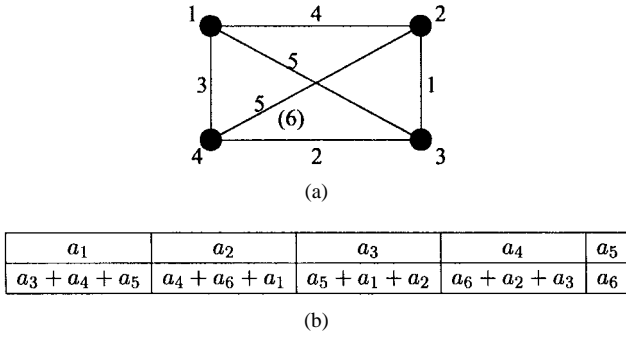
Since the construction of  $\hat{B}_{2n}$  can be obtained from  $\hat{B}_{2n+1}$  simply by deleting the last parity column, here we focus on the graph description of  $\hat{B}_{2n+1}$ .  $\hat{B}_{2n+1}$  has  $2n$  information bits and  $n(2n - 1)$  parity bits, which can be represented exactly by a complete graph of  $2n$  vertices,  $K_{2n}$ , which also has exactly  $\binom{2n}{2} = n(2n - 1)$  edges. The mapping is simple: one information bit can be represented by one vertex, and the parity bit that is the sum of two information bits can be represented by the edge that links the two corresponding vertices. So the only remaining problem is to define on  $K_{2n}$  the grouping relation that determines which information and parity bits occupy the same column of the code. This can be thought of as labeling the vertices and edges of the complete graph  $K_{2n}$  in such a way that information bit and parity bits in the same column are labeled with the same label. Since  $\hat{B}_{2n+1}$  has  $2n + 1$  columns, we need  $2n + 1$  labels. Notice that the each of the first  $2n$  columns has exactly one information bit and  $n - 1$  parity bits, and that the last column has  $n$  parity bits. A formal way of describing the  $\hat{B}_{2n+1}$  is as follows.

#### Description 1: Graph Description of $\hat{B}_{2n+1}$

Given a complete graph  $K_{2n}$  with  $2n$  vertices, which are labeled with integers from 1 to  $2n$ , find an edge-labeling scheme such that

- 1) each edge is labeled exactly once by an integer from 1 to  $2n + 1$
- 2) For any pair of vertices  $(i, j)$  and any other vertex  $k$ , where  $i, j, k \in [1, 2n]$ , there is always a path to  $k$  from either  $i$  or  $j$ , using only the edges labeled with  $i$  or  $j$ .
- 3) For any vertex  $i$  and any other vertex  $k$ , where  $i, k \in [1, 2n]$ , there is always a path from  $i$  to  $k$ , using only the edges labeled with  $i$  or  $2n + 1$ .

With the above description, it is easy to see that the vertex and edges with the label  $i$  in the  $K_{2n}$  represent the information bit and parity bits in the  $i$ th column of  $\hat{B}_{2n+1}$ . The properties 2) and 3) ensure that any two columns of the code can recover the information bits in all other columns, thus the code is of column distance  $2n$ . Fig. 5 shows such a labeling of  $K_4$  and the corresponding  $\hat{B}_5$ , where  $a_1$  through  $a_4$  are the information bits.

Fig. 5. (a) graph and (b) array representations of  $\hat{B}_5$ .Fig. 6. (a) Graph and (b) array representations of  $B_5$ .

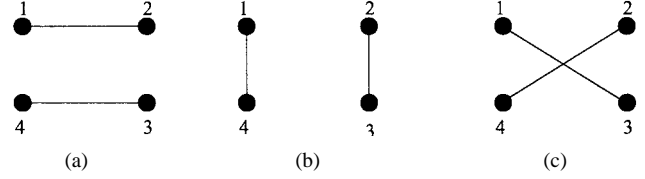
Naturally, if the edges of  $K_{2n}$  are used to represent information rather than parity bits, and vertices to represent the parity bits, it should be expected that by using such a labeling scheme and reindexing the edges, such a complete graph can represent  $B_{2n+1}$ , i.e., the B-Code itself. And in fact this is true. In the graph representation of  $B_{2n+1}$ , a parity bit is the sum of all the information bits whose edges are incident with its vertex.  $B_{2n}$  can easily be obtained from  $B_{2n+1}$  by setting all the information bits in the last column to zero and then deleting them after the parity bits are changed accordingly.  $B_5$  is shown in Fig. 6, where the edge labeled with (6) represents the information bit  $a_6$  in the fifth column. It is also interesting to point out that  $B_5$  happens to be a *perfect* code too, i.e., it achieves the Hamming bound [16].

### III. B-CODE AND P1F

As already described in Section II, constructing the B-Code amounts to the same problem as designing an edge-labeling scheme such as in Description 1 for a complete graph  $K_{2n}$ . Fortunately this can be related to another graph theory problem, namely, the perfect one-factorization problem.

#### A. Perfect One-Factorization of Complete Graphs

**Definition 3** [15]: Let  $G = (V, E)$  be a graph. A *factor* or *spanning subgraph* of  $G$  is a subgraph with vertex set  $V$ . In particular, a *one-factor* is a factor which is a regular graph of degree 1. A *factorization* of  $G$  is a set of factors of  $G$  which are pairwise *edge disjoint*, and whose union is all of  $G$ . A *one-factorization* of  $G$  is a factorization of  $G$  whose factors are all one-factors. In particular, a one-factorization is *perfect*

Fig. 7. (a)–(c) are three one-factors, that together form a perfect one-factorization of  $K_4$ .

if the union of any pair of its one-factors is a *Hamilton cycle*, a cycle that passes through every vertex of  $G$ .

Fig. 7 shows a perfect one-factorization of  $K_4$ . A perfect one-factorization of  $K_6$  is shown in Fig. 8(b), where edges with the same label form a one-factor.

The perfect one-factorization of complete graphs has been studied for many years since its introduction in [11]. It was first shown in [1] that

**Theorem 2:** If  $p$  is an odd prime, then  $K_{p+1}$  and  $K_{2p}$  have perfect one-factorizations.

Constructions of P1F for  $K_{p+1}$  and  $K_{2p}$  were also given in [1]. An alternative construction of P1F for  $K_{2p}$  was proven in [10] to be isomorphic to the construction in [1]. Additionally, constructions of P1F for  $K_{2n}$ 's whose  $n$ 's are some other sporadic integers have also been found [14], [15]. However, it still remains a conjecture [14], [15] that

**Conjecture 1:** For any positive integer  $n$ ,  $K_{2n}$  has perfect one-factorization(s).

#### B. Equivalence Between the B-Code and P1F

Let  $P_{2n+2}$  be a P1F for  $K_{2n+2}$ . Recall that  $\hat{B}_{2n+1}$  has  $2n+1$  columns, and  $P_{2n+2}$  also has  $2n+1$  one-factors. So, if we are able to find a one-to-one mapping between the columns and one-factors, then we can get constructions for  $\hat{B}_{2n+1}$  from  $P_{2n+2}$ , and *vice versa*. Luckily enough, such a mapping does exist. The following two algorithms give such a one-to-one mapping.

**Algorithm 1:** Constructing  $\hat{B}_{2n+1}$  from  $P_{2n+2}$ .

- Step 1. Label the vertices of  $K_{2n+2}$  with  $0, 1, \dots, 2n, \infty$ ;
- Step 2. If a P1F exists for  $K_{2n+2}$ , then let  $F_i$  denote the one-factor which contains the edge  $0i$ , where  $i = 1, 2, \dots, 2n, \infty$ ;
- Step 3. In each  $F_i$ , delete the two vertices  $0$  and  $\infty$  and all the edges which are incident with either of them; For  $i = 1, 2, \dots, 2n$ , label all the remaining edges in  $F_i$  with  $i$ , and label all the remaining edges of  $F_\infty$  with  $2n+1$ .

Fig. 8 shows the construction of  $\hat{B}_5$  from  $P_6$ , where in Fig. 8(a)  $\infty$  is replaced with 5, and the edges with the same label  $i$  form the one-factor  $F_i$ .

**Theorem 3:** Algorithm 1 gives a graph as described in Description 1, i.e., a construction of  $\hat{B}_{2n+1}$ .

**Proof:** First observe that in a P1F of  $K_{2n+2}$ , each edge appears exactly once in the whole set of the one-factors, thus Step 2) is feasible. Now check the conditions of the graph description of  $\hat{B}_{2n+1}$ .

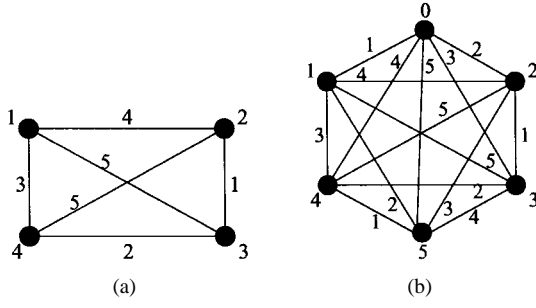


Fig. 8. Constructing (a)  $\hat{B}_5$  from (b)  $P_6$ .

- 1) Obviously holds.
- 2) Since  $F_i$  and  $F_j$  ( $i, j \in [1, 2n]$ ) are two one-factors of a  $P_{2n+2}$ , their union is a Hamilton cycle of  $2n+2$  vertices, after deleting the vertices 0 and  $\infty$  and the four edges incident with them, the Hamilton cycle breaks into two paths, covering all the remaining vertices from 1 to  $2n$ . These paths start from  $i$  or  $j$ , thus this condition holds.
- 3) Since  $F_i$  and  $F_\infty$  ( $i \in [1, 2n]$ ) form a Hamilton cycle of  $2n+2$  vertices where  $0\infty$  is an edge, after the deletion of the vertices 0 and  $\infty$  and the three edges incident with them, the Hamilton cycle becomes one path starting from  $i$ , thus this condition holds.  $\square$

Since  $B_{2n+1}$  and  $\hat{B}_{2n+1}$  can be described with the same complete graph  $K_{2n}$ , both  $B_{2n+1}$  and  $\hat{B}_{2n+1}$  can be constructed from  $P_{2n+2}$ . Additionally,  $B_{2n}$  and  $\hat{B}_{2n}$  can be easily obtained from  $B_{2n+1}$  and  $\hat{B}_{2n+1}$ , so the B-Code and its dual (of size  $n \times l$ ) can be constructed from the known P1F constructions of  $K_{2n+2}$ . In particular, from Theorem 2

**Theorem 4:** For any odd prime  $p$ , a B-Code and its dual code of size  $n \times l$  can be constructed, where  $n$  is either  $(p-1)/2$  or  $p-1$ .

When  $n = (p-1)/2$ , the corresponding B-Code is the code in [18] and [5]. The B-Code of  $n = p-1$  was not known before. Using Algorithm 1,  $B_{2p-1}$  (and, of course,  $B_{2p-2}$ ) can be obtained from  $P_{2p}$ , a P1F for  $K_{2p}$ . Here we describe a construction of P1F for  $K_{2p}$ , where  $p$  is an odd prime [10].

**Algorithm 2:** Construction of P1F for  $K_{2p}$  ( $p$  is an odd prime).

Let  $V = \{v_0, v_1, \dots, v_{2p-1}\}$  be the vertex set of the complete graph  $K_{2p}$ , then for any integer with  $0 \leq s \leq 2p-1$  and  $s \neq p$ , construct a 1-factor  $G_s$  of  $K_{2p}$  as follows:

$$G_s = \begin{cases} \{v_i v_j | i+j = s, i \not\equiv j \pmod{2p}\} \cup \{v_{s/2} v_{(s/2)+p}\}, & \text{if } s \text{ is even} \\ \{v_i v_j | i: \text{odd}, i-j = s \pmod{2p}\}, & \text{otherwise} \end{cases}$$

where the subscript of  $v$  is expressed modulo  $p$ . Then the set of  $G_s$  denoted by

$$GN_{2p} = \{G_s | 0 \leq s \leq 2p-1, s \neq p\}$$

is a P1F for  $K_{2p}$ .

The next natural question is: Can we get  $P_{2n+2}$  from  $B_{2n+1}$ ? The answer is yes and the following algorithm can do it.

**Algorithm 3:** Constructing  $P_{2n+2}$  from  $\hat{B}_{2n+1}$ .

- Step 1. If  $\hat{B}_{2n+1}$  exists, use Description 1 of  $\hat{B}_{2n+1}$ , let  $\tilde{F}_i$  denote the set of edges with the label  $i$ , where  $i = 1, 2, \dots, 2n$ , and let  $\tilde{F}_\infty$  denote the set of the edges with the label  $2n+1$ ;
- Step 2. Add two vertices 0 and  $\infty$  to  $K_{2n}$ ;
- Step 3. For  $i = 1, 2, \dots, 2n$  and  $\infty$ , add the edges  $i0$  and  $k\infty$  to  $\tilde{F}_i$ , where  $k$  is an integer from 1 to  $2n$  such that the expanded set  $F_i$  is a one-factor of the complete graph  $K_{2n+2}$  of vertices  $0, 1, 2, \dots, 2n, \infty$ .

**Theorem 5:** Algorithm 3 gives a  $P_{2n+2}$ .

*Proof:* Observe that because of the MDS and optimal encoding properties of  $\hat{B}_{2n+1}$ , in each of the first  $2n$  columns of  $\hat{B}_{2n+1}$

- 1) each information bit appears at most once;
- 2) there is exactly one bit which does not appear; also no pair of the columns miss the same bit, since otherwise that bit cannot be recovered from just these two columns;
- 3) in the last column, each bit appears exactly once.

Thus 2) guarantees that in Step 3, there exists a unique  $k$ . Further, 1) ensures that for any pair of columns  $i$  and  $j$ , where  $i, j = 1, 2, \dots, 2n$ , the two vertices  $i$  and  $j$  can only be the endpoints of the two paths in the graph description. Thus Step 3 of the above algorithm makes the union of any pair of  $F_i$  and  $F_j$ , where  $i, j = 1, 2, \dots, 2n$ , into a Hamilton cycle. Step 3 also makes the union of any  $F_i$  ( $i = 1, 2, \dots, 2n$ ) and  $F_\infty$  a Hamilton cycle. Thus  $\{F_1, F_2, \dots, F_{2n}, F_\infty\}$  is a P1F of  $K_{2n+2}$ , i.e., it is  $P_{2n+2}$ .  $\square$

Theorems 2 and 5 reveal a surprising result:

**Theorem 6:** Constructing  $\hat{B}_{2n+1}$  (or equivalently  $B_{2n+1}$ ) is equivalent to constructing  $P_{2n+2}$ , i.e.,  $P_{2n+2} \Leftrightarrow B_{2n+1}$ .

Note that the equivalence does not include the B-Codes of even length, i.e.,  $B_{2n}$ . This equivalence, however, already shows that any progress in P1F gives a new B-Code, and *vice versa*.

### C. Erasure Decoding of the B-Code

Obviously, the encoding of the B-Code can be done using Algorithm 1. Now consider erasure decoding for the B-Code. Recall that the dual B-Code can recover all information bits from any two columns. Erasure decoding for the dual B-Code is almost obvious from its graph description (Description 1). The two paths, starting from  $i$  and  $j$  and leading to all the other vertices in the graph, give the decoding chain used in recovering a  $\hat{B}$ -Code from its  $i$ th and  $j$ th columns. Fig. 9 shows the decoding chain used in recovering  $\hat{B}_5$  from its first column and its second, third, and fifth columns, respectively.

The B-Code can recover any two missing columns. Decoding for the B-Code itself is almost the same as for its dual, except that the roles of edges and vertices are exchanged. Fig. 10 shows the decoding chains for recovering  $B_5$ 's first column and second, third, and fifth columns, respectively. Comparing the decoding sequences here with those of  $\hat{B}_5$ , it is easy to observe that the decoding chains for recovering

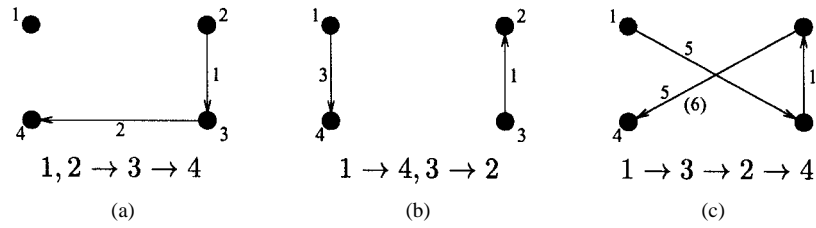


Fig. 9. Erasure decoding of  $\hat{B}_5$ : recovering from its first and (a) second, (b) third, and (c) fifth columns. The decoding chains for each case are also listed. 1 through 4 are the information bits in the corresponding columns.

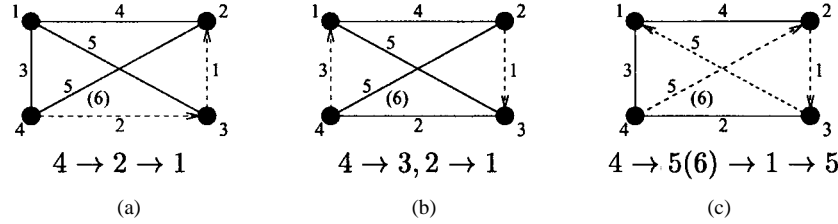


Fig. 10. Erasure decoding of  $B_5$ : recovering its first and (a) second, (b) third, and (c) fifth columns. The decoding chains for each case are also listed. 1 through 6 are the information bits in the corresponding columns, except that 6 is also in the fifth column.

the  $i$ th and  $j$ th columns of  $B_l$  are just the reversed sequences of those for recovering its dual code  $\hat{B}_l$  from its  $i$ th and  $j$ th columns. This also shows that the two codes are dual to each other, since their graph descriptions are dual to each other.

Finally, the above decoding algorithms can be summarized as follows.

*Algorithm 4:* Erasure decoding of the dual B-Code.

The edges labeled  $i$  and  $j$  create two paths which span the vertex set. Starting at vertex  $i$  and at vertex  $j$ , use these paths, by adding a known information bit and a known parity bit to recover a new unknown information bit. Repeating this step along each path recovers the dual B-code from its  $i$ th and  $j$ th columns.

*Algorithm 5:* Erasure decoding of the B-Code.

To recover the  $i$ th and  $j$ th columns of the B-Code, use the same paths with edges labeled with  $i$  and  $j$ . This time, traverse the paths in the opposite directions of the corresponding paths for the dual B-Code. Along each path, add all known information and parity bits to get a new unknown information bit. Repeating this step along each path recovers the  $i$ th and  $j$ th columns of the B-Code from the other  $n - 2$  columns.

#### D. Error Decoding of the B-Code

Recall that a B-Code of size  $n \times l$  is of distance 3, so it can correct one error. To do this, the key is again to locate the error location; the error value can easily be determined once the location is found. One way to find the error location is to make a table that maps syndromes to single-error locations, and then do a table lookup after calculating the syndrome of a received array. The drawbacks are 1) such a table is needed for each B-Code and 2) table lookup is not efficient in both computation time and space (since the total number of 1-error syndromes is  $2^n$ ). Another rather straightforward algorithm is to consider the  $i$ th column and  $(i+1)$ th column to be erasures (where  $i = 1, 3, 5, \dots, 2n-1$  for  $l = 2n$ ; if  $l = 2n+1$ , the  $l$ th column and the 1st column are also included), and then recover those columns. If exactly one of the recovered columns

differs from the original ones, then that discrepant column is the error column. This algorithm can correct one error. The algorithm requires on average  $n/2$  erasure decodings, each of which needs  $2n(l-3)$  additions, thus the average total number of additions is  $n^2(l-3)$ , which is in the order of  $n$  times of  $n \times l$ . Another shortcoming is that the algorithm will give a false decoding result if more than one error occur.

We present here a more efficient decoding algorithm for correcting one error. Observe the relation between a B-Code and its dual from their graph descriptions: if an information bit  $i$  of a B-Code appears in a set  $P$  of parity-bit positions, then in its dual code, the elements of  $P$  will be information bits and  $i$  will be then a parity bit; further, all the elements of  $P$  appear in the parity bit  $i$ . Thus if there is a single column error in a received array of a B-Code, use the syndrome of this received array as the information vector of the dual code. In the obtained dual codeword, the parity bits in the error column of the B-Code should be zero, while other parity bits are nonzero because of the structural properties of the B-Code observed in the proof of Theorem 5. This differentiates the error location from other columns. The decoding algorithm can be described semi-formally as follows.

*Algorithm 6:* Error Decoding of the B-Code.

- 1) Given a received array  $R$  of size  $n \times l$ , calculate its syndrome, denoted as  $S$  (which is a vector of length  $2n$ );
- 2) If  $S$  is a zero vector, then the received array  $R$  is a codeword of  $B_l$ ; otherwise, go to next step;
- 3) Use  $S$  as the information vector of the dual B-Code  $\hat{B}_l$ , encode to get a codeword  $C$  of  $\hat{B}_l$ ;
- 4) If the weight of the syndrome  $S$  is even, and if there is a unique all-zero column in  $C$ , then this is the error column of the original received array  $R$ . On the other hand, if the weight of syndrome  $S$  is odd, and if there is a unique column whose information bit is nonzero and whose parity bits are all-zero, then this is the error column of  $R$ ;

- 5) If the error column of  $R$  is found in the above step, regard this column as an erasure and recover it; otherwise, declare decoding failure: there are at least two error columns in  $R$ .

Notice that the above property holds only when the B-Code is defined in  $G(2^m)$ , i.e., each cell of the B-Code consists of a block of  $m$  binary bits. However, this decoding algorithm can be modified to work for the general B-Codes defined in  $G(q^m)$ , where  $q$  is not a power of 2. Here we will stick to the case where  $q = 2$  and  $m = 1$ .

Before we prove the correctness of the above algorithm, we show an example of it.

*Example 1: Error correcting for the B-Code.*

Consider  $B_6$ , whose graph description is shown in Fig. 1. (Its array description is easy to get from either its graph description or the array description of  $\hat{B}_6$ , as in Fig. 1.) If two received arrays are as follows:

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then the two syndromes are, respectively,

$$S_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Since neither  $S_1$  nor  $S_2$  is a zero vector, both  $R_1$  and  $R_2$  have errors. Now use  $S_1$  and  $S_2$  as information vectors of  $\hat{B}_6$ , whose graph and array descriptions are shown in Fig. 1. We get two codewords of  $\hat{B}_6$

$$C_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Since the weight of  $S_1$  is even, and column 1 is the unique all-zero column in  $C_1$ , column 1 is the error column of  $R_1$ ; on the other hand, since the weight of  $S_2$  is odd, and column 1 is the unique column in  $C_2$  whose information bit is nonzero and whose parity bits are all zeros, column 1 is the error column of  $R_2$  too. Once the error column of  $R_1$  ( $R_2$ ) is found, the error value is easy to get. The corrected arrays of  $R_1$  and  $R_2$  are both all-zero arrays.  $\square$

Now we prove the correctness of the decoding algorithm.

*Proof:* The following can be observed from the graph description of the B-Code: each information bit appears in exactly two parity bits, and this information bit and the two parity bits are in three different columns; in addition, two information bits from the same column cannot appear in the same parity bit, thus all possible errors in the information bits of a single column contribute even weight to its syndrome vector. On the other hand, single parity-bit error adds exactly one to the weight of the syndrome vector, i.e., a parity bit makes the weight of the syndrome odd.

Suppose there is only one error column in a received array of a B-Code. Call this column the  $i$ th column. Consider the following two cases:

Case 1: *All errors occur in information bits.* Then the syndrome should be of even weight. Now we prove that the  $i$ th column of the obtained codeword of the dual B-Code is the only all-zero column.

- 1) *The  $i$ th column is an all-zero column.* This is true because of the relation between the B-Code and its dual: in the B-Code, an information bit  $i$  appears in two parity bits  $P_1$  and  $P_2$ , and in the dual B-Code these two bits  $\hat{P}_1$  and  $\hat{P}_2$  are information bits, and the bit  $\hat{i}$  is a parity bit such that both  $\hat{P}_1$  and  $\hat{P}_2$  appear in  $\hat{i}$ . Since  $P_1 = P_2 = i$  and  $\hat{P}_1 = \hat{P}_2$ ,  $\hat{i} = 0$ . Thus all parity bits of the  $i$ th column of the dual B-Code are zero. Additionally, since there is no error in the parity bit of the B-Code, the information bit of the  $i$ th column of the dual B-Code is also zero. So the entire  $i$ th column of the dual B-Code is zero.
- 2) *All the other columns are nonzero columns.* If there were at least one more all-zero column in the codeword, then the weight of the codeword would be no greater than  $l - 2$ , which contradicts the fact the minimum distance of the dual B-Code is  $l - 1$ . Here  $l$  is the length of the codeword.

Case 2: *An error also occurs in the parity bit as well.* In the dual B-Code, among all the columns which have both an information bit and parity bits (if the length of the dual B-Code is odd, there is one column containing no information bit), by the linearity of the dual B-Code, the  $i$ th column has a nonzero information bit, and all its parity bits are zero. No other column can have a nonzero information bit and all-zero parity bits. The reason is as follows: the weight of the information bits in the dual B-Code is *odd*, and all the information bits appear in the  $i$ th column. Since each information bit is missing from exactly one of the columns, the number of nonzero information bits that appear in the  $j$ th column ( $j \neq i$ ) is even. Thus if the information bit of the  $j$ th column is nonzero, then at least one of its parity bits is nonzero, since each parity bit is the sum of two information bits, and the total number of information bits which appear in the parity bits is now *odd*.

When multiple column errors occur, there can be multiple all-zero columns or multiple columns with the first component nonzero and all other components zero. This concludes the proof for the correctness of the decoding algorithm.  $\square$



The complexity of the above decoding algorithm is easy to analyze. For a received array of size  $n \times l$ , the syndrome calculation requires  $2(l-2)n$  additions; the encoding of the dual B-Code requires  $(l-2)n$  additions; finally, correcting one erasure requires  $(l-3)n$  additions. This adds up to  $(4l-9)n$  additions, which is linear in the number of total bits in an array of size  $n \times l$ . The same trick used here cannot be applied directly to correct multiple errors for the dual B-Code, since multiple errors can weave together and cannot be easily separated. In general, it still remains a challenge to correct multiple errors efficiently (total additions linear in total number of bits in an array) for array codes.

#### IV. FURTHER EQUIVALENCE DISCUSSION

The equivalence between the B-Code and P1F has been shown in the preceding section. It is quite clear that  $B_{2n}$  can be constructed from  $B_{2n+1}$  simply by shortening, namely, setting all the information bits in the last column to zero. Similarly,  $\hat{B}_{2n}$  can be derived from  $\hat{B}_{2n+1}$  by puncturing, i.e., deleting the last parity row. The relations among  $P_{2n+2}$ ,  $B_{2n+1}$ , and  $B_{2n}$  can be described as follows, where  $\Rightarrow$  means to lead to:

$$P_{2n+2} \Leftrightarrow B_{2n+1} \Rightarrow B_{2n}.$$

A further question is whether  $B_{2n+1}$  (or  $\hat{B}_{2n+1}$ ) can be constructed from a known construction of  $B_{2n}$  (or  $\hat{B}_{2n}$ ), i.e., whether the last  $\Rightarrow$  can be replaced with  $\Leftrightarrow$ . Our conjecture is yes.

*Conjecture 2:* For any positive integer  $n$ ,  $\hat{B}_{2n+1}$  (or  $B_{2n+1}$ ) can be constructed from  $\hat{B}_{2n}$  (or  $B_{2n}$ ) using Algorithm 7.

*Algorithm 7:* Constructing  $\hat{B}_{2n+1}$  from  $\hat{B}_{2n}$

Extend a given  $\hat{B}_{2n}$  by adding one more column, which contains, as parity bits, all the unused or unlabeled edges in the graph description of  $\hat{B}_{2n}$ .

The B-Codes shown in Figs. 1, 5, and 6 are all what we call *shift codes*, and it is easy to verify Conjecture 2 is true for these examples.

*Definition 4 (Shift Code):* An array code (of size  $n \times l$ ) is called a *shift code* if any row of its parity-check matrix is just a cyclic shift of the first row, i.e., the remaining columns of the code can be constructed by cyclically shifting the first column.

In general, for a shift B-Code, Conjecture 2 can be proven true, namely,

*Theorem 7:* For any shift B-Code,  $\hat{B}_{2n+1}$  (or  $B_{2n+1}$ ) can be constructed from  $\hat{B}_{2n}$  (or  $B_{2n}$ ) using Algorithm 7.

*Proof:* Given a shift dual B-Code  $\hat{B}_{2n}$ , notice that the missing edges are the diagonals,  $(i, i+n)$  (addition is modulo  $2n$ ). Indeed, if  $(i, i+n)$  were present in column  $j$  of  $\hat{B}_{2n}$ , then it would be included in column  $n+j$  as well, because of the shift property, making the code non-MDS.

To complete the proof, we need to show that by using an arbitrary column  $j$  of  $\hat{B}_{2n}$  together with the diagonals  $(i, i+n)$ ,  $0 \leq i < n$ , one can recover all remaining  $2n-1$  columns, i.e., we indeed have  $\hat{B}_{2n+1}$ . Suppose that is not true.

There exists a column  $j$  in which a set of edges, combined with the diagonals, form a loop

$$a_1 + n + a_2 + n + \cdots + a_q + n = 0 \pmod{2n}$$

where  $q$  is the number of edges involved in the loop,  $a_i$ 's are their lengths, and  $n$  is the length of the diagonals. For example, let  $n = 6$ ,  $q = 3$ ,  $a_1 = 1$ ,  $a_2 = 2$ , and  $a_3 = 3$

$$1 + 6 + 2 + 6 + 3 + 6 = 24 = 0 \pmod{12}.$$

We will show that this cannot happen. We will show that column  $j+n$  and column  $j$  form a loop and, therefore, the original code is not  $\hat{B}_{2n}$ . Using the above equation:

$$\sum_{i=1}^q a_i = qn.$$

Because column  $j+n$  is a cyclic shift of column  $j$ , it contains a set of edges of lengths  $b_i = a_i$  such that  $A_1$  connects to  $B_2$ , which in turn connects to  $A_3$ , etc.,

$$\sum_{i=1}^q a_i + \sum_{i=1}^q b_i = 2qn = 0 \pmod{2n}.$$

There is a loop. □

If Conjecture 2 can be proven true for any arbitrary B-Code, then we can get a *strong* equivalence between the B-Codes and P1F, i.e., the B-Code construction is completely equivalent to the P1F construction.

#### V. CONCLUSIONS

In this paper we have presented the B-Code and its dual, a new class of optimal MDS array codes of size  $n \times l$  (where  $l = 2n$  or  $2n+1$ ) with distance 3 (or  $l-1$  for the dual code). We proved an equivalence between the B-Code and perfect one-factorizations using a new graph description of the B-Code. We also described encoding and decoding algorithms for the B-Code and its dual based on their graph descriptions. There are a number of open problems: i) are the B-Code constructions strongly equivalent to perfect one-factorizations? ii) can the graph description of the B-Codes be extended to design optimal array codes of arbitrary distance? iii) how does one efficiently correct multiple errors for the dual B-Code (or other array codes)? and the ultimate question, iv) can coding theory techniques be used to solve the P1F conjecture?

#### ACKNOWLEDGMENT

We are grateful to Dr. M. Blaum of IBM Almaden Research Center for his helpful discussions with us.

#### REFERENCES

- [1] B. A. Anderson, "Symmetry groups of some perfect 1-factorizations of complete graphs," *Discr. Math.*, vol. 18, pp. 227–234, 1977.
- [2] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. Comput.*, vol. 44, pp. 192–202, Feb. 1995.
- [3] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Trans. Inform. Theory*, vol. 42, pp. 529–542, Mar. 1996.

- [4] M. Blaum and R. M. Roth, "New array codes for multiple phased burst correction," *IEEE Trans. Inform. Theory*, vol. 39, pp. 66–77, Jan. 1993.
- [5] ———, "On lowest-density MDS codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 46–59, Jan. 1999.
- [6] M. Blaum, P. G. Farrell, and H. C. A. van Tilborg, "Chapter on array codes," in *Handbook of Coding Theory*, V. S. Pless and W. C. Huffman, Eds. Amsterdam, The Netherlands: North-Holland, 1998, pp. 1855–1909.
- [7] P. G. Farrell, "A survey of array error control codes," *ETT*, vol. 3, no. 5, pp. 441–454, 1992.
- [8] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [9] R. M. Goodman, R. J. McEliece, and M. Sayano, "Phased burst error correcting arrays codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 684–693, 1993.
- [10] M. Kobayashi, "On perfect one-factorization of the complete graph  $K_{2p}$ ," *Graphs Combinatorics*, vol. 5, pp. 351–353, 1989.
- [11] A. Kotzig, "Hamilton graphs and Hamilton circuits," in *Theory of Graphs and Its Applications (Proc. Symp.)* (Smolenice), 1963, pp. 63–82.
- [12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
- [13] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
- [14] D. G. Wagner, "On the perfect one-factorization conjecture," *Discr. Math.*, vol. 104, pp. 211–215, 1992.
- [15] W. D. Wallis, *One-Factorizations*. Norwell, MA: Kluwer, 1997.
- [16] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
- [17] L. Xu and J. Bruck, "X-Code: MDS array codes with optimal encoding," *IEEE Trans. Inform. Theory*, vol. 45, pp. 272–276, Jan. 1999.
- [18] G. V. Zaitsev, V. A. Zinov'ev, and N. V. Semakov, "Minimum-check-density codes for correcting bytes of errors, erasures, or defects," *Probl. Inform. Transm.*, vol. 19, no. 3, pp. 197–204, 1983.